Lista de Exercícios - Prof. Paulo Sérgio Lopes de Souza

1. Decodifique as seguintes instruções de máquina do MIPS 0x000B5080, 0x000A40C2, 0x36110008 (em hexadecimal) e determine o que elas fazem. Considere os seguintes conteúdos (base 10) nos registradores: $t0=10 , $t1=1000, $t2=64, $t3=32, $t4=8000, $s0=1, $s1=256, $s2=15, $s3=7 e $s4=31. Qual será o conteúdo de cada registrador citado acima depois da execução de cada instrução separadamente (não em sequência)?

2. Considere a seguinte instrução do Computador XYZ, a qual realiza a carga de um operando vindo da memória principal: ***load $r1, 1000***. Qual será o conteúdo do registrador $r1 depois de executada tal instrução se a mesma estiver usando: (a) modo de endereçamento imediato, (b) modo de endereçamento direto e (c) modo de endereçamento indireto. Para resolver essa questão, considere que a memória possui o seguinte conteúdo (valores na base 10):

[end]	Conteúdo
…	…
1000	1200
…	…
1200	1204
1204	1208
1208	1000

3. Considere o Código em C e o Código em Assembly do MIPS abaixo. Preencha os campos vazios no Código em Linguagem de Máquina (tabela abaixo) com os valores corretos na base 10:

Código em C	Código em Assembly para o MIPS		
While (save[i] == k) i = i + j;	Loop:	add $t1, $s3, $s3	# $t1 = 2 * i
		add $t1, $t1, $t1	# $t1 = 4 * i
		add $t1 $t1, $s5	# $t1 = save[i]
		lw $t0, 0($t1)	# $t0 = save[i]
		bne $t0, $s5, Exit	# desvia para Exit se sve[i] <> k
		add $s3, $s3, $s4	# i = i + j
		j Loop	# desvia para Loop
	Exit:		

[end]	Instruções em Linguagem de Máquina para o MIPS					
80000	0	19	19	9	0	32
80004	0	9	9	9	0	32
80008	0	9	22	9	0	32
80012	35	9	8	0		
80016	5	8	21			
80020	0	19	20	19	0	32
80024	2					
80028	…					
80012	35	9	8	0		

4. Converta de Assembly MIPS para Binário ou de Binário para Assembly MIPS, conforme o caso, as instruções solicitadas abaixo. Expresse as instruções binárias em hexadecimal, quando for o caso, na coluna mais à direita. Expresse as instruções em Assembly, quando for o caso, na coluna "Instruções em Assembly". A primeira coluna indica o endereço de memória (em decimal) que a instrução se encontra.

End.Byte na RAM	Instruções em Assembly	Instrução em Binário	Instrução em Binário, representada em Hexadecimal
8			0xAFB00000
20	beq $s0, $zero, nao		
36	nao: addi $v0, $zero, 10		
44			0x01000008

5. Converta o código abaixo de binário MIPS para assembly MIPS. Os números antes de cada linha representam os endereços de memória em decimal que as instruções estariam. Converta todo o código e na sequência escreva, apenas em poucas palavras, o que este código faz.

```
            .data
            .align 0
    s:      .asciiz "abccccdcc"
    c:      .ascii "c"
            .text
            .align 2
            .globl main
    main:                   ####    espaço para a conversão das instruções.    ####
```

000- la $a0, s _____

004- la $t0, c _____

008- 0x81050000 _____

012- 0x0c00000a _____

016- move $t0, $a0 _____

020- 0x80440000 _____

024- li $v0, 1 _____

028- 0x0000000c _____

032- li $v0, 10 _____

036- 0x0000000c _____

040- 0x23bdfff8 _____

044- 0xafbf0004 _____

048- 0xafb00000 _____

052- 0x00801020 _____

056 0x80500000 _____

060- 0x12000003 _____

064- 0x12050003 _____

068- 0x20420001 _____

072- 0x0800000e _____

076- 0x20020000 _____

080- 0x8fbf0004 _____

084- 0x8fb00000 _____

088- 0x23bd0008 _____

092- 0x03e00008 _____

096- ...

Escreva aqui o que este código faz.

_____

_____

Não descreva os passos do algoritmo. Escreva o que ele faz. Em tese, apenas uma palavra já seria o suficiente. Pode usar mais de uma palavra, mas não escreva mais que as duas linhas acima.

6. Um computador com palavras de 32 bits possui uma memória com endereçamento a Byte. O char usa 8 bits para o padrão ASCII. O short int usa 16 bits e o int usa 32 bits. Inclua na memória, a partir do endereço 0 (zero), o conteúdo da struct:

```
struct {
        char c1[3];       // o conteúdo(ascii) é: 'A', 'B', 'C'
        short int si;     // o conteúdo é: 0x0123
        int a;            // o conteúdo é:  0x456789AB
} estrutura;
```

Para a inclusão do conteúdo da *struct*, considere em (a) uma Memória *Big-Endian* e em (b) uma Memória *Little-Endian*. Os números em decimal na figura abaixo indicam o endereço do Byte da célula de memória. Os dados do tipo ***char*** devem ser representados na memória como caracteres: ***'A'***, ***'B' e ´C´***. Os dados do tipo ***short int*** e ***int*** devem ser representados na memória em hexadecimal.

(a) Memória RAM Big-Endian

				<byte>
8	9	10	11	< ***ender.***>
				<byte>
4	5	6	7	< ***ender.***>
				<byte>
0	1	2	3	< ***ender.***>

(b) Memória RAM Little-Endian

8	9	10	11
4	5	6	7
0	1	2	3